Liquid Pouring Monitoring via Rich Sensory Inputs

Tz-Ying Wu[1,⋆], Juan-Ting Lin[1,⋆], Tsun-Hsuang Wang[1], Chan-Wei Hu[1],
Juan Carlos Niebles[2], and Min Sun[1]

[1] Department of Electrical Engineering, National Tsing Hua University, Taiwan
{gina9726, brade31919, johnsonwang0810, huchanwei1204}@gmail.com,
sunmin@ee.nthu.edu.tw
[2] Department of Computer Science, Stanford University, USA
jniebles@cs.stanford.edu

Abstract. Humans have the amazing ability to perform very subtle manipulation task using a closed-loop control system with imprecise mechanics (i.e., our body parts) but rich sensory information (e.g., vision, tactile, etc.). In the closed-loop system, the ability to monitor the state of the task via rich sensory information is important but often less studied. In this work, we take liquid pouring as a concrete example and aim at learning to continuously monitor whether liquid pouring is successful (e.g., no spilling) or not via rich sensory inputs. We mimic humans' rich sensories using synchronized observation from a chest-mounted camera and a wrist-mounted IMU sensor. Given many success and failure demonstrations of liquid pouring, we train a hierarchical LSTM with late fusion for monitoring. To improve the robustness of the system, we propose two auxiliary tasks during training: inferring (1) the initial state of containers and (2) forecasting the one-step future 3D trajectory of the hand with an adversarial training procedure. These tasks encourage our method to learn representation sensitive to container states and how objects are manipulated in 3D. With these novel components, our method achieves $\sim 8\%$ and $\sim 11\%$ better monitoring accuracy than the baseline method without auxiliary tasks on unseen containers and unseen users respectively.

Keywords: Monitoring Manipulation, Multimodal Fusion, Auxiliary Tasks.

1 Introduction

Researchers in cognitive science community have conducted several studies [1,2] of mental simulation, and proved that humans have some internal mechanisms to reason daily life physics with relative ease. Some robotics research borrows a hand from human demonstrations to tackle manipulation problems; for example, recently, Edmonds et al. [3] leverage multimodal sensor to capture poses and contact forces to learn the manipulation of opening medicine bottles. Humans

[⋆] indicates equal contribution

can be viewed as closed-loop control systems with imprecise mechanics (i.e., our body parts) but rich sensory information (e.g., vision, tactile, etc.). The sensory feedback helps us continuously reason the environment, and plan our next action according to it. In the closed-loop system, the ability to monitor the state of the task via rich sensory information is important but often less studied. Monitoring subtle manipulation task is useful for both in-home elder care system and virtual training in medical scenarios (e.g., training surgical operation), since a system with this kind of ability can further assist people to accomplish subtle tasks.

Liquid pouring is a subtle manipulation task that humans learn during childhood and can easily perform on a daily basis. This task requires continuously monitoring environmental states such as the liquid level in containers and the relative position and motion between containers in order to adjust future actions toward not spilling. For instance, if the receiver container is empty and the source container is tilting slowly, one should speed-up the tilting action. In contrast, if the receiver container is almost full and the source container is tilting fast, one should slow down the tilting action to prevent overflow. This suggests that both object states, relative position and motion are very important cues for subtle manipulation tasks such as liquid pouring. With the ability to monitor liquid pouring, an intelligent system can either stop the user from spilling, or bring a duster to the user when the liquid is spilled.

Monitoring liquid pouring activity is a very subtle task compared to mainstream activity recognition tasks such as action classification or temporal detection [4,5]. Hence, only a few works have made progress toward this direction in computer vision. Alayrac et al. [6] propose to discover object states and manipulation actions in videos. However, they only consider empty versus full (binary) container states and multiple discrete actions where pouring is one of them. Recently, Mottaghi et al. [7] propose to reason about volume and content in liquid containers to predict how much liquid will remain in the container if we tilt it by x degrees (referred to as pouring prediction). However, we argue that such prediction target has limited application since it does not directly answer how to pour liquid successfully or whether the pouring action results in success or failure.

In this work, we take liquid pouring as a concrete example and aim at learning to continuously monitor whether liquid pouring is successful (e.g., not spilling) or not via rich sensory inputs. Cognitive scientists suggest that people have the ability to simulate pouring behaviors in their mind, which is mentioned in [1]. However, there remain discrepancies between the simulation and the real results. By continuously observe current environmental states, people can adjust their ways to manipulate the object (e.g. the angle of the container) in order to reach their goal. This process can be viewed as a closed-loop control. In order to borrow a hand from humans' physical reasoning ability, we mimic humans' rich sensors using synchronized observation from a chest-mounted camera and a wrist-mounted IMU sensor as the input (details in section 5). The target output for monitoring is a binary class: a success or a failure pouring trial. To study liquid pouring monitoring in the real world by leveraging human demonstrations,

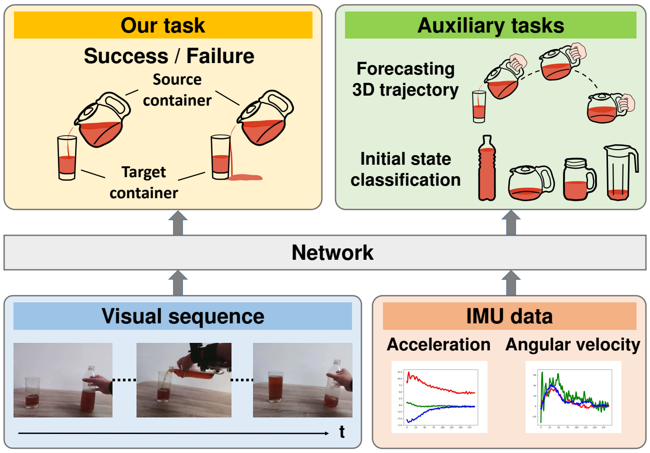


Fig. 1. Overview. From a series of visual observations and IMU data, our model can monitor if this sequence is a success or failure with two auxiliary tasks: initial object state classification (different containers with different initial liquid levels) to ensure the recurrent model encode states sensitive features; forecasting 3D trajectory requires the ability to model hand dynamics during the pouring process, providing a strong cue for our monitoring task. The details of auxiliary tasks are described in section 4

we collect a liquid pouring dataset containing both successful and failed demonstrations with all inputs and outputs information mentioned above. To the best of our knowledge, this is the first dataset with multimodal sensor information for studying monitoring in a subtle liquid pouring task.

Given many success and failure demonstrations of liquid pouring, we train a hierarchical LSTM [8] with late fusion to incorporate rich sensories inputs without significantly increasing the model parameters as compared to early fusion models. To further improve the generalizability of our method, we introduce two auxiliary tasks during training: (1) predicting the initial state of containers and (2) forecasting the one-step future 3D trajectory of the hand with an adversarial training procedure. These auxiliary tasks encourage our method to learn representation sensitive to container states and how objects are manipulated in 3D. In our experiments, our method achieves $\sim 8\%$ and $\sim 11\%$ better monitoring accuracy than the baseline method without auxiliary tasks on unseen containers and unseen users respectively.

2 Related Work

Activity Recognition. Activity recognition has received lots of attention from the computer vision community and already has many released datasets [9,10,4,5,11] containing diverse actions. Many prior works on activity recognition focus on understanding human activity through observing body poses [12,13,14], scenes

[15,16] or objects interacting with human [17,18,19,20]. There are also many works [21,22,23] considering recognizing activity through egocentric videos, some of which use depth sensor [24,25] as well in attempt to enhance the perception of the changes in the environment. There are also methods [26] and datasets [27,28] utilizing multimodal sensor inputs to perform activity recognition. These established datasets mainly focus on diverse activity recognition and do not include failure cases. However, we focus more on distinguishing subtle differences among behaviors targeting on the same objective (liquid pouring). Therefore, we collect our own liquid pouring dataset with multimodal sensor data which includes both success and failure cases (details in section 5).

Fine-grained activity recognition. Many methods focused on interacting and manipulating motions between human and objects. Lei *et al.*[25] applied RGB-D camera to achieve the robust object and action recognition. There are also methods utilizing spatiotemporal information [29,30,31,32,33]. By combining spatiotemporal and object semantic features, Yang *et al.*[29] find key interaction without using further object annotations. In this work, rather than designing special procedures to mine unique spatiotemporal features, we introduce auxiliary tasks to learn feature good for multiple tasks.

Environmental State Estimation. In liquid pouring sequences, container and the liquid state can be estimated from RGB inputs. Alayrac *et al.*[6] model the interaction between actions and objects in a discrete manner. Some methods further demonstrate that liquid amount can be estimated by combining semantic segmentation CNN and LSTM [34,7]. In contrast, our main goal is not to explicitly recognize environmental states. We aim at implicitly learning environmental state sensitive features such that our performance in monitoring can be improved. Recently, Sermanet *et al.* [35] also propose to learn states sensitive feature in a self-supervised manner.

Robot Liquid Pouring. In the robotics community, there are a number of works [36,37,38,39,40,41,42] directly tackle the manipulating task of liquid pouring without considering the monitoring task. [36] build a liquid dynamic model using optical flow. [41,42] are developed in synthetic environments. Tamosiunaite *et al.* [37] apply model-based reinforcement learning. Rozo *et al.* [38] propose a parametric hidden Markov model to direct regress control commands. Brandl *et al.* [39] learn to generalize pouring to unseen containers by warping the functional parts of the unseen containers to mimic the functional parts of a seen container. Schenck and Fox [40] propose to first estimate the volume of liquid in a container; then, a simple PID controller is used to pour specific amounts of liquid. However, all of the methods above are not evaluated on generalization jointly across users, containers states, container instances.

3 Overview

In this section, we first formulate the problem of monitoring liquid pouring. Next, we describe our recurrent model for fusing multimodal data. Our method with two auxiliary tasks will be mainly described in section 4.

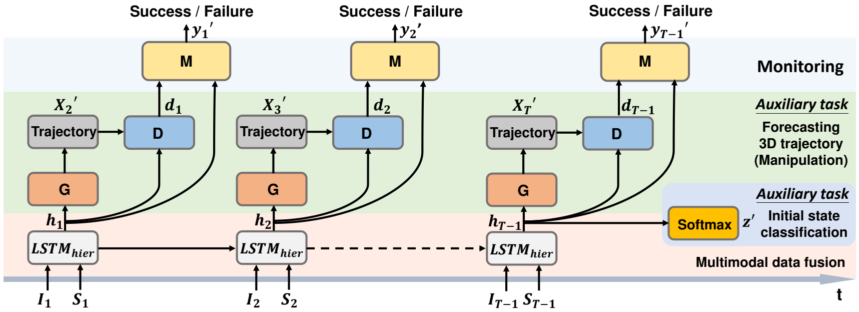


Fig. 2. Model architecture. Our model consists of a hierarchical LSTM $LSTM_{hier}$ (details in subsection 3.2), a generator G, a discriminator D and a monitoring module M (details in section 4). There are two auxiliary tasks in our method, which are 3D trajectory forecasting (green shading) and initial state classification (blue shading). At each time step t, $LSTM_{hier}$ will encode visual observation I_t and IMU data S_t to h_t (red shading). G will generate a trajectory X'_{t+1} according to hidden encoding h_t. D will distinguish if the input trajectory is generated or not corresponding to h_t, which models the dynamics during the manipulation. M will predict if this pouring sequence is a success or failure based on the discriminator score d_t and hidden encoding h_t. At the end of the sequence, the model will classify 36 initial states as an auxiliary task

3.1 Problem Formulation

Notations. For all of our notations, general font style stands for ground truth data, and prime stands for predictions. For example, y_t is the ground truth label for whether the sequence is a success and y'_t is the prediction. Notations with boldface denote a sequence of data. t denotes a certain time step, and T stands for the total time steps of the sequence.

Observation. To capture visual and motion information like liquid content, container type and dynamics of the demonstrator's hand during the pouring process, we use a multimodal sensing system including a camera on the front chest and an IMU sensor on the wrist. At each time step t, the camera observes visual observation I_t, and the 6DOF IMU sensor captures motion observation $S_t = \{\mathbf{a}^1, \mathbf{a}^2, ..., \mathbf{a}^N\}$, where $\mathbf{a}^i$ is the i'th sample in the current time step, $i \in 1 \sim N$, and N denotes the number of samples in this time step. In practice, $N = 38$, i.e., IMU sensor will capture 38 samples within two consecutively captured camera frames. $\mathbf{a} = \{a_1, a_2, a_3, a_4, a_5, a_6\}$ is a single piece of real-valued data from the IMU, where (a_1, a_2, a_3) is the acceleration and (a_4, a_5, a_6) is the angular velocity corresponding to x, y, and z axis. Simultaneously, at each time step t, we obtain hand 3D trajectory ground truth $X_t = (P, R)$ by a HTC Vive tracker mounted on the wrist, where $P = (p_x, p_y, p_z)$ and $R = (r_x, r_y, r_z)$ stand for the position part and rotation part in world coordinate respectively. Note that HTC Vive system is only used in training.

Goal. In our task, we aim at learning to monitor whether the pouring liquid sequence is a success or failure with two auxiliary tasks, which are initial object

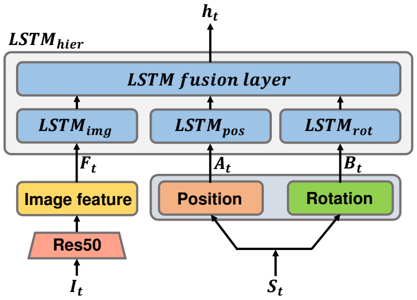


Fig. 3. LSTM encoder. Our hierarchical LSTM encoder $LSTM_{hier}$ consists of 3 LSTM cells ($LSTM_{img}$, $LSTM_{pos}$, $LSTM_{rot}$) at the first level and a LSTM fusion layer to fuse these hidden encodings at the second level, fusing multimodal inputs containing image feature F_t, hand position feature A_t and hand rotation feature B_t computed from IMU sensor

state classification (IOSC) and next-step hand 3D trajectory forecasting (TF). Considering the input sequence containing visual images $\mathbf{I} = \{I_1, I_2, ..., I_T\}$ and IMU data $\mathbf{S} = \{S_1, S_2, ..., S_T\}$, the output of our model for each time step t are the prediction y'_t indicating whether the sequence is a success for our monitoring task and next-step trajectory prediction X'_{t+1} for 3D trajectory forecasting, where $t \in 1 \sim T-1$, T denotes the total time steps of the sequence. In the end of the whole sequence, our model will predict the initial object state z' of the sequence among the 36 variations (details in section 5).

3.2 Multimodal Data Fusion

To catch and combine the temporal sequence of input from image and IMU sensor, we adopt a hierarchical LSTM proposed by [8] to handle scale differences among multimodal inputs. In the first layer of our module $LSTM_{hier}$ (see Figure 3), there are 3 LSTM cells ($LSTM_{img}$, $LSTM_{pos}$, $LSTM_{rot}$) with different hidden layer sizes to encode the inputs from three different sources: (1) image feature $F_t = \mathbf{Res50}(I_t)$ extracted from the pool5 layer of ResNet50 [43] with dimension of 1×2048, (2) hand position feature: the aggregation of acceleration along 3 axis $A_t = \{(a_1^i, a_2^i, a_3^i)\}_{i=1}^N \subset S_t$ with dimension of $1 \times 3N$ and (3) hand rotation feature: the aggregation of angular velocity along 3 axis $B_t = \{(a_4^i, a_5^i, a_6^i)\}_{i=1}^N \subset S_t$ with dimension of $1 \times 3N$. Then the encoded features are concatenated as the input to the second layer consisting of a single LSTM cell. The output encoded feature $h_t = LSTM_{hier}(F_t, A_t, B_t)$ of the hierarchical LSTM will be passed to the generator G, discriminator D and the monitor module (please refer to section 4).

4 Monitoring with Auxiliary Tasks

Monitoring the success of a pouring sequence is a challenging task since subtle changes in states of the environment are hard to perceive. Intuitively, the initial object state and the hand dynamics are the strong cues for monitoring pouring process. We model the object and manipulator (i.e., hand) states implicitly by a hierarchical LSTM $LSTM_{hier}$ and introduce two auxiliary tasks, 3D trajectory forecasting (TF) and initial object state classification (IOSC). In this section, we describe the details of the two auxiliary tasks and our monitoring module.

4.1 Forecasting 3D Trajectory

Forecasting 3D trajectory is a path for us to learn to model the dynamics of the manipulator during the pouring sequence. The most naive way to predict trajectory is to train direct regression on demonstration sequences; however, the generated trajectory will be very limited to the data distribution of training data as the amount and the diversity of training data is limited. To model the distribution of successful demonstration and to generate more diverse trajectories, we introduce adversarial training loss L_{adv} proposed by Goodfellow *et al.* [44] here with a generator G to generate trajectory prediction and a discriminator D to distinguish if the input trajectory is generated or not (see Figure 2).

Generator. Taking the encoded feature h_t from $LSTM_{hier}$ as input, our generator predicts next-step trajectory $X'_{t+1} = G_{\theta_G}(h_t)$ as output, where G_{θ_G} is a three-layer fully-connected feed-forward network parametrized by θ_G. Our generator has two objectives:

(1) Generate the trajectory which is close to the ground truth demonstration. (modeled by the regression loss). (2) Fool discriminator with the generated trajectory (modeled by the adversarial loss).

Thus, our loss function for the generator can be derived as follows,

$$L_{Gen} = L_{reg} + \lambda * L_{adv}, \tag{1}$$

where λ is the weighting between the two different losses (we empirically set λ to 1), L_{reg} is the regression loss, and L_{adv} stands for the adversarial loss.

The regression loss is defined as follows,

$$L_{reg} = \frac{1}{T-1} \sum_{t=1}^{T-1} dist(X_{t+1}, G_{\theta_G}(h_t)), \tag{2}$$

where $dist()$ is the distance function, X_{t+1} is the ground truth trajectory, $G_{\theta_G}(h_t)$ is the generated trajectory, and T denotes the total time steps of the sequence. Recall the trajectory X_{t+1} is composed of two parts, position $P = (p_x, p_y, p_z)$ and rotation $R = (r_x, r_y, r_z)$; likewise $G_{\theta_G}(h_t) = (P', R')$, where $P' = (p'_x, p'_y, p'_z)$, $R' = (r'_x, r'_y, r'_z)$.

The distance function is defined as

$$dist(X_{t+1}, G_{\theta_G}(h_t)) = MSE(P, P') + \sum_{k=x,y,z} (1 - \cos(r_k - r'_k)), \tag{3}$$

where MSE denotes Mean Squared Error. Here we use different distance metrics for rotation and translation because adopting cosine distance in angular difference is more reasonable. In particular, the cosine distance between 359° and 0° is small, but its mean square error is large. Note that we empirically adopt the same weighting for the position loss and rotation loss since the effect of different weightings is marginal on the performance.

The adversarial loss is defined as follows,

$$L_{adv} = \frac{1}{T-1} \sum_{t=1}^{T-1} -\log D_{\theta_D}(h_t, G_{\theta_G}(h_t)), \tag{4}$$

where D_{θ_D} is the discriminator of our model and will be elaborated later.
Discriminator. In training time, the discriminator takes both the encoded feature at that time step h_t and the predicted trajectory $X'_{t+1} = G_{\theta_G}(h_t)$ from the generator or ground truth trajectory X_{t+1} as inputs with the objective of catching generated trajectory from the generator. Adopting similar design from the generator, our discriminator D_{θ_D} is also modeled with a three-layer fully-connected feed-forward network parameterized by θ_D. The discriminator loss is defined as follows,

$$L_{Dis} = \frac{1}{T-1}\sum_{t=1}^{T-1}[-\log\left(D_{\theta_D}(h_t, X_{t+1})\right) - \log(1 - D_{\theta_D}(h_t, G_{\theta_G}(h_t)))] \tag{5}$$

In testing time, given the encoded feature h_t and generated trajectory X'_{t+1} of the certain time step t, the discriminator will predict the score $d_t = D_{\theta_D}(h_t, X'_{t+1}), t \in 1 \sim T-1$ of whether the input sequence is generated or not.

4.2 Initial Object State Classification

As we mention above, hand motion and initial object states are the two strong cues for monitoring pouring sequences. Learning the embedding of the data sequence is critical since the amount of training data is limited. To learn a good representation for monitoring, we train the classification on the initial object state based on the hidden encoding from the hierarchical LSTM $LSTM_{hier}$ in the end of each successful demonstration sequence (see Figure 2) as follows,

$$q = \text{Softmax}(\theta_q, h_{T-1}), \tag{6}$$

$$z' = \arg\max_{c \in \mathcal{Z}} q(c), \tag{7}$$

$$L_{cls} = -\log q(z), \tag{8}$$

where h_{T-1} is the hidden encoding at the last time step of the sequence, θ_q is the parameter of the classifier and $q \in R^{|\mathcal{Z}|}$ is the softmax probability of initial object states in $\mathcal{Z}$. z' is the prediction of the initial object state and z denotes the ground truth initial object state. In our case, $|\mathcal{Z}| = 36$, which means there are 36 variations of initial object states (details can be referred to section 5).

4.3 Monitoring Module

We propose a monitoring module M, which is designed as a single-layer network to predict whether a pouring sequence is a success or not given the hidden representation h_t from $LSTM_{hier}$ and the discriminator score d_t as inputs (see Figure 2). The output of the monitoring module is defined as,

$$y'_t = \text{M}_{\theta_M}(h_t, d_t), \tag{9}$$

where θ_M is the parameter of M and y'_t is the prediction of success or failure. We train our monitoring module with cross-entropy loss. The architecture of our monitoring module is compact and effective since our model has already learned powerful feature that can capture the appearance changes and hand dynamics during the pouring process through auxiliary tasks.

4.4 Implementation Details

We use ResNet50[43] trained on ImageNet[45] as the visual feature extractor. The input size of $LSTM_{img}$ is 2048, and the input size of both $LSTM_{pos}$ and $LSTM_{rot}$ are $3N$ ($N = 38$ in our case). $LSTM_{img}$ hidden size is 512, and both $LSTM_{pos}$ and $LSTM_{rot}$ hidden size are 128. The second layer of hierarchical LSTM has its hidden size 512. Generator G and discriminator D are the 3-layered fully-connected network with each layer of size 128. Monitor module is a fully-connected layer of size 256. We train our model for 3000 epochs with batch size 24. Learning rate is $1e^{-4}$. We optimize all objectives with equal weightings.

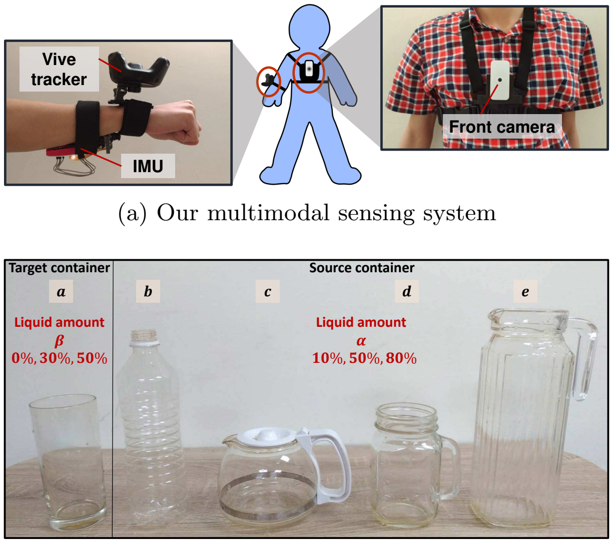
5 Dataset

In order to examine our method on monitoring whether the pouring sequence belongs to successful / failure sequences, we collect both successful and failure pouring sequences with our multimodal sensing system. We have one chest-mounted camera to capture the first-person view observation; one wrist-mounted 6DOF IMU sensor and one tracker of the HTC Vive motion tracking system on the right wrist to catch both the motion observation and the ground truth trajectory simultaneously. Figure 4.a is the illustration of the devices on the demonstrator. We illustrate how we collect different kinds of demonstrations below.

Variations of pouring sequences. Our single pouring sequence consists of pouring liquid from the source container with initial liquid amount α to target container with β amount of liquid. Similar to [7], we roughly divide the container states into discrete labels. In successful sequences, the demonstrator tries to fill target container with the liquid in the source container without spilling out any liquid. If target container is filled to about 80% full, the demonstration will stop even if there is still liquid left in the source container. For single demonstrator, we will record the demonstrations with different kinds of containers and different initial liquid amounts to obtain more diverse demonstrations. For source container, we use 4 different containers b, c, d, e in Figure 4.b with three different initial liquid amount α: {10%, 50%, 80%}. We use container a in Figure 4.b as the target container with three different initial liquid amount β: {0%, 30%, 50%}. Combining the different settings in source container, α, and β, we can obtain total 36 different initial object states. In practice, we will record 5 repeated sequences for each initial object state setting. As a result, for a single demonstrator, we can obtain 180 demonstration sequences.

Pouring styles. In addition to different variations in the liquid amount and container appearances, we collect demonstrations conducted by 5 different demonstrators to ensure the diversity in pouring styles from person to person.

Failure sequences. In general, there can be many ways to conduct a failure



(a) Our multimodal sensing system

(b) Variations of initial settings

Fig. 4. Settings to collect our dataset. (a) A camera is mounted on the chest to capture visual images. On the wrist, there is a vive tracker and an IMU sensor. (b) We use these containers to create variations of initial settings (details in section 5)

sequence. However, to model the monitoring tasks, we choose one of the most common mistakes made by humans during the pouring sequences: *Spill out* (The demonstrator accidentally spill out some liquid during the pouring action.) Regarding the variations and pouring styles, we use the same settings from the successful sequences: (1) 5 repeated sequences for each of 36 variations. (2) 5 different demonstrators to ensure diverse pouring behaviors. Hence, the total amount of demonstration is $2 * 5 * 5 * 36 = 1800$.

6 Experiments

In this section, we introduce the evaluation metrics and settings used in our experiments. We then describe our monitoring experiments and discuss our experimental results with ablation studies.

6.1 Metrics

In our experiments, we observe that prediction varies a lot across users and thus, to eliminate bias introduced by specific users, we evaluate our model in a leave-one-out cross-validation fashion using the following metrics:
Success/Failure accuracy — metric for monitor task. It shows how well the model discriminates a successful pouring sequence from a failed one. It directly indicates the performance of our main task.

Classification accuracy — metric for initial object state classification. It shows how well the model recognizes what kinds of container and amount of liquid in the containers in a pouring sequence.
Regression error — metric for trajectory forecasting. It is the error between 6-dimensional 3D trajectories recorded by HTC Vive and predicted 3D trajectories. Note that due to distinct properties of position and rotation error, the two errors are calculated separately.

6.2 Setting Variants

To study the effectiveness of each independent component in our network, we evaluate different settings described below in the following experiments.
Vanilla RNN: Our fusion RNN without auxiliary tasks. The model is a LSTM encoder (see subsection 3.2) followed by fully-connected layers. The fully-connected layers perform success/failure classification based on the encoded features.
RNN w/ IOSC: Our fusion RNN with an auxiliary task, initial object state classification (IOSC). The details of IOSC are described in subsection 4.2.
RNN w/ TF: Our fusion RNN with an auxiliary task, trajectory forecasting (TF). The details of TF are described in subsection 4.1.
Ours w/o adv.: Our fusion RNN with two proposed auxiliary tasks, initial object state classification and trajectory forecasting. In this setting, we treat one-step trajectory forecasting as a regression task (see Equation 2).
Ours: Our fusion RNN with two proposed auxiliary tasks, initial object state classification and trajectory forecasting. In this setting, we introduce the adversarial training loss (see Equation 4) to generate more diverse trajectories.

6.3 Monitoring Liquid Pouring

We consider 3 scenarios to test our method's generalization ability. Firstly, we assume that our model is used to monitor a specific group of users with a specific set of containers. Then, in a more challenging scenario, we assume the model need to monitor unseen containers as well. Finally, we consider that the model needs to monitor unseen users. More details are described below.
Cross Trial Experiment. This experiment is the most simple case. Models are trained and tested on data of the same group of users with the same container set, but training data and testing data are collected from different trials of pouring. In this easiest scenario, success/failure classification poses minor challenge here and is well solved. From Table 1, we can see that our method generates better performance on monitoring than the baseline method (i.e. *vanilla RNN*), which lacks two auxiliary tasks.
Cross Container Experiment. This is a common scenario that may occur in the real use case. When using different containers to pour liquid, the whole pouring sequences may be very different. For instance, there are huge changes in the appearance and the pouring trajectories between the case of the teapot and the bottle. We run leave-one-out cross-validation on the 4 different source containers to test whether our model can generalize to unseen containers. The initial states

are only related to the liquid amount in the source (10%, 50%, 80%) and target container (0%, 30%, 50%), so we have 9 initial states (rather than 36 states) in total. The results in Table 2 show that our method achieves better performance on monitoring than the baseline method, since it successfully catches the change of states and the hand dynamics during the pouring sequence.

Cross User Experiment. This is the most challenging scenario, since different demonstrators may have very different pouring styles. Considering a specific set of containers, models are trained on data of 4 different users and tested on 1 user other than the 4 users in training set. The main difference among cross-user data is the variance in pouring styles. To be more precise, this experiment examines generalization ability in IMU sensor data sequences. By looking at success/failure accuracy shown in Table 3, we can find that both auxiliary tasks, initial-state classification and trajectory forecasting, brings considerable improvement in monitoring object manipulation. From Figure 6, we can observe that our model's prediction correctly follows the visual cues. Initial object state classification helps the model know what the source container and the target container are, and the amount of liquid in both containers. Trajectory forecasting helps the model learn local dynamics of pouring sequences. Remarkably, by comparing our method and *Ours w/o adv.*, we can find that adversarial training introduced in our method significantly boosts initial state classification and slightly improves trajectory forecasting. From the results, we infer that there is implicitly-shared knowledge between the two auxiliary tasks and a more robust trajectory forecasting may enhance initial state classification. Adversarial training does help regarding obtaining a better understanding of pouring behaviors and increase the performance of our model in monitoring task.

6.4 Discussion

In this section, we further discuss each component in our network and the future feasibilities. Firstly, we do ablation study on LSTM architecture under the cross-user scenario, comparing the hierarchical LSTM (see subsection 3.2) to a 2-layer LSTM. The latter one is an early fusion method that data from different modalities is directly concatenated together and fed into the 2-layer LSTM. The results in Table 4 show that the hierarchical LSTM with late fusion outperforms the naive 2-layer LSTM in all tasks and this may be due to the capability of the hierarchical LSTM to handle scale difference and imbalanced dimension among multimodal inputs.

Secondly, we study the effect of the adversarial loss to the whole network. Recall that we introduce adversarial loss since there are multiple feasible trajectories for each data sample. However, these errors assume that there is only one truth position and rotation of each testing sample. As mentioned above, our model learns a more general concept and will predict trajectory based on common knowledge considering pouring, whereas prediction of "*Ours w/o adv.*" heavily relies on knowledge of seen trajectories and will drastically fail if testing pouring sequences have little in common with training data. This can be observed in Figure 5.a. Also, the adversarial loss will allow the model to generate

Table 1. The results of cross trial experiments

	succ./fail. acc.	classification acc.	position error	rotation error
Vanilla RNN	99.65 %	*N/A*	*N/A*	*N/A*
Ours w/o adv.	100 %	96.50 %	0.020 *m*	7.58°
Ours	100 %	96.07 %	0.020 *m*	**6.80**°

Table 2. The results of cross container experiments

	succ./fail. acc.	classification acc.	position error	rotation error
Vanilla RNN	89.16 %	*N/A*	*N/A*	*N/A*
Ours w/o adv.	96.45 %	63.92 %	0.040 *m*	11.11°
Ours	**97.11** %	**67.69** %	**0.038** *m*	11.30°

Table 3. The results of cross user experiments

	succ./fail. acc.	classification acc.	position error	rotation error
Vanilla RNN	81.95 %	*N/A*	*N/A*	*N/A*
RNN w/ IOSC	89.25 %	68.51 %	*N/A*	*N/A*
RNN w/ TF	90.82 %	*N/A*	0.033 *m*	14.15°
Ours w/o adv.	92.97 %	64.15 %	0.033 *m*	14.20°
Ours	**93.25** %	**75.69** %	0.033 *m*	**14.06**°

Table 4. Ablation study on LSTM architecture

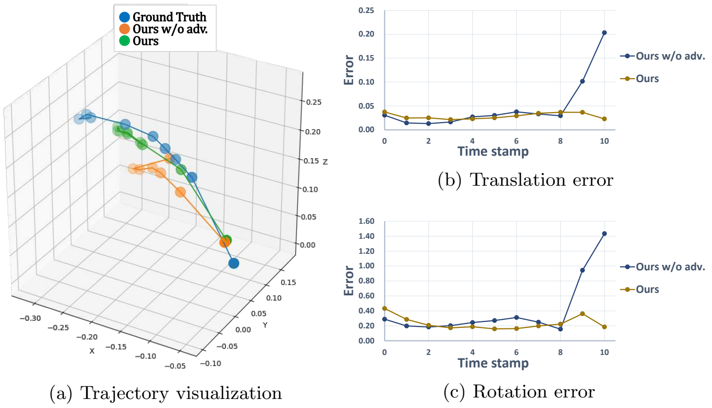
LSTM architecture	succ./fail. acc.	classification acc.	position error	rotation error
2-*layer*	87.06 %	58.92 %	0.033 *m*	14.72°
hierachical	**93.25** %	**75.69** %	0.033 *m*	**14.06**°

more diverse trajectories, which means the model will observe more diverse hidden states in later steps. The trajectory forecasting errors in Figure 5.b and 5.c show that "*Ours*" and "*Ours w/o adv.*" have comparable errors at early steps, but the former one perform better in later steps.

Our experiments show that introducing auxiliary tasks is beneficial for understanding the subtle liquid pouring task. By implicitly modeling the environmental states and hand dynamics, we improve liquid pouring monitoring significantly. We believe the general idea applies to other subtle manipulating tasks like opening doors, driving nails and cutting bread. Intuitively speaking, opening doors also involves mapping visual (e.g., what types of doors) and non-visual (e.g., hand motion) observations into environmental states to facilitate monitoring whether the door is opened. Monitoring different tasks may need different auxiliary tasks to make use of rich sensories in order to learn both visual and non-visual signals.

7 Conclusion

In this work, we aim at learning to monitor whether liquid pouring is successful (e.g., not spilling) or not using synchronized visual and IMU signals. We propose a novel method containing two auxiliary tasks during training: inferring (1) the



(a) Trajectory visualization

(b) Translation error

(c) Rotation error

Fig. 5. Trajectory forecasting comparison between "*Ours w/o adv.*" and "*Ours*". (a) Ground truth, "*Ours w/o adv.*" and "*Ours*" are shown in blue, orange and green, respectively. Time is visualized as color intensity goes from dark to light. Apparently, "*Ours w/o adv.*" failed to forecast the trajectory at a later stage of liquid pouring, while "*Ours*" can still follow the trend. (b)(c) "*Ours*" and "*Ours w/o adv.*" have comparable errors at early steps, but the former one performs better in later steps

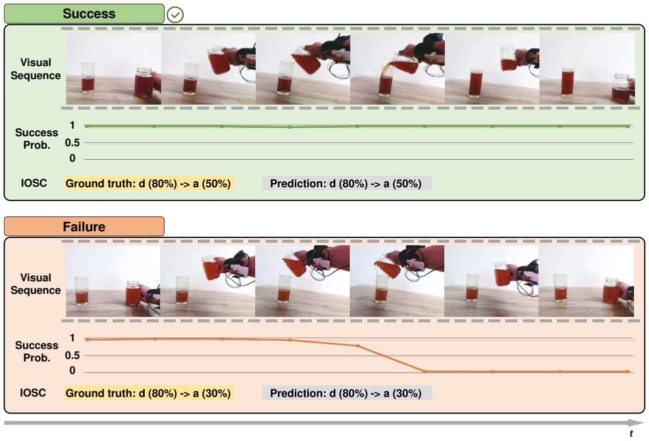

Success
Visual Sequence
Success Prob.
1
0.5
0
IOSC
Ground truth: d (80%) -> a (50%)
Prediction: d (80%) -> a (50%)
Failure
Visual Sequence
Success Prob.
1
0.5
0
IOSC
Ground truth: d (80%) -> a (30%)
Prediction: d (80%) -> a (30%)
t

Fig. 6. Monitoring along time. The prediction correctly follows the visual cues

initial state of containers and (2) forecasting the one-step future 3D trajectory of the hand with an adversarial training procedure. These tasks encourage our method to learn representation sensitive to container states and how objects are manipulated in 3D. On our newly collected liquid pouring dataset, our method achieves $\sim 8\%$ and $\sim 11\%$ better monitoring accuracy than the baseline method without auxiliary tasks on unseen containers and unseen users respectively.

Acknowledgements We thank Stanford University for collaboration. We also thank MOST 107-2634-F-007-007, Panasonic and MediaTeK for their support.

References

1. Kubricht, J., Jiang, C., Zhu, Y., Zhu, S.C., Terzopoulos, D., Lu, H.: Probabilistic simulation predicts human performance on viscous fluid-pouring problem. CogSci (2016)
2. Bates, C.J., Yildirim, I., Tenenbaum, J.B., Battaglia, P.W.: Humans predict liquid dynamics using probabilistic simulation. CogSci (2015)
3. Edmonds, M., Gao, F., Xie, X., Liu, H., Qi, S., Zhu, Y., Rothrock, B., Zhu, S.C.: Feeling the force: Integrating force and pose for fluent discovery through imitation learning to open medicine bottles. IROS (2017)
4. Abu-El-Haija, S., Kothari, N., Lee, J., Natsev, P., Toderici, G., Varadarajan, B., Vijayanarasimhan, S.: Youtube-8m: A large-scale video classification benchmark. arXiv:1609.08675 (2016)
5. Heilbron, F.C., Escorcia, V., Ghanem, B., Niebles, J.C.: Activitynet: A large-scale video benchmark for human activity understanding. In: CVPR. (2015)
6. Alayrac, J.B., Sivic, J., Laptev, I., Lacoste-Julien, S.: Joint discovery of object states and manipulating actions. In: ICCV. (2017)
7. Mottaghi, R., Schenck, C., Fox, D., Farhadi, A.: See the glass half full: Reasoning about liquid containers, their volume and content. In: ICCV. (2017)
8. Nishida, N., Nakayama, H.: Multimodal gesture recognition using multi-stream recurrent neural network. In: PSIVT. (2015)
9. Soomro, K., Zamir, A.R., Shah, M.: Ucf101: A dataset of 101 human actions classes from videos in the wild. arXiv:1212.0402 (2012)
10. Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., Serre, T.: Hmdb: a large video database for human motion recognition. In: ICCV. (2011)
11. Gu, C., Sun, C., Ross, D.A., Vondrick, C., Pantofaru, C., Li, Y., Vijayanarasimhan, S., Toderici, G., Ricco, S., Sukthankar, R., Schmid, C., Malik, J.: Ava: A video dataset of spatio-temporally localized atomic visual actions. In: CVPR. (2018)
12. Rohrbach, M., Amin, S., Andriluka, M., Schiele, B.: A database for fine grained activity detection of cooking activities. In: CVPR. (2012)
13. Chéron, G., Laptev, I., Schmid, C.: P-cnn: Pose-based cnn features for action recognition. In: ICCV. (2015)
14. Jhuang, H., Gall, J., Zuffi, S., Schmid, C., Black, M.J.: Towards understanding action recognition. In: ICCV. (2013)
15. Vu, T.H., Olsson, C., Laptev, I., Oliva, A., Sivic, J.: Predicting actions from static scenes. In: ECCV. (2014)
16. Zhang, Y., Qu, W., Wang, D.: Action-scene model for human action recognition from videos. (2014)
17. Moore, D.J., Essa, I.A., Hayes, M.H.: Exploiting human actions and object context for recognition tasks. In: ICCV. (1999)
18. Delaitre, V., Sivic, J., Laptev, I.: Learning person-object interactions for action recognition in still images. In: NIPS. (2011)
19. Gupta, A., Kembhavi, A., Davis, L.S.: Observing human-object interactions: Using spatial and functional compatibility for recognition. TPAMI (2009)
20. Gupta, A., Davis, L.S.: Objects in action: An approach for combining action understanding and object perception. In: CVPR. (2007)
21. Fathi, A., Rehg, J.M.: Modeling actions through state changes. In: CVPR. (2013)
22. Bambach, S., Lee, S., Crandall, D.J., Yu, C.: Lending a hand: Detecting hands and recognizing activities in complex egocentric interactions. In: ICCV. (2015)

23. Ma, M., Fan, H., Kitani, K.M.: Going deeper into first-person activity recognition. In: CVPR. (2016)
24. Hu, J.F., Zheng, W.S., Lai, J., Zhang, J.: Jointly learning heterogeneous features for rgb-d activity recognition. In: CVPR. (2015)
25. Lei, J., Ren, X., Fox, D.: Fine-grained kitchen activity recognition using rgb-d. In: UbiComp. (2012)
26. Song, S., Cheung, N.M., Chandrasekhar, V., Mandal, B., Liri, J.: Egocentric activity recognition with multimodal fisher vector. In: Acoustics, Speech and Signal Processing (ICASSP), IEEE (2016)
27. de la Torre, F., Hodgins, J.K., Montano, J., Valcarcel, S.: Detailed human data acquisition of kitchen activities: the cmu-multimodal activity database (cmu-mmac). In: CHI Workshop. (2009)
28. Roggen, D., Calatroni, A., Rossi, M., Holleczek, T., Förster, K., Tröster, G., Lukowicz, P., Bannach, D., Pirkl, G., Ferscha, A., et al.: Collecting complex activity datasets in highly rich networked sensor environments. In: INSS, IEEE (2010)
29. Zhou, Y., Ni, B., Hong, R., Wang, M., Tian, Q.: Interaction part mining: A mid-level approach for fine-grained action recognition. In: CVPR. (2015)
30. Zhou, Y., Ni, B., Yan, S., Moulin, P., Tian, Q.: Pipelining localized semantic features for fine-grained action recognition. In: ECCV. (2014)
31. Peng, X., Zou, C., Qiao, Y., Peng, Q.: Action recognition with stacked fisher vectors. In: ECCV. (2014)
32. Sun, S., Kuang, Z., Sheng, L., Ouyang, W., Zhang, W.: Optical flow guided feature: A fast and robust motion representation for video action recognition. In: CVPR. (2018)
33. Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y., Paluri, M.: A closer look at spatiotemporal convolutions for action recognition. In: CVPR. (2018)
34. Schenck, C., Fox, D.: Detection and tracking of liquids with fully convolutional networks. In: RSS workshop. (2016)
35. Sermanet, P., Lynch, C., Hsu, J., Levine, S.: Time-contrastive networks: Self-supervised learning from multi-view observation. arXiv:1704.06888 (2017)
36. Yamaguchi, A., Atkeson, C.G.: Stereo vision of liquid and particle flow for robot pouring. Humanoids (2016)
37. Tamosiunaite, M., Nemec, B., Ude, A., Wrgtter, F.: Learning to pour with a robot arm combining goal and shape learning for dynamic movement primitives. IEEE-RAS (2011)
38. Rozo, L., Jimnez, P., Torras, C.: Force-based robot learning of pouring skills using parametric hidden markov models. In: 9th International Workshop on Robot Motion and Control. (2013)
39. Brandi, S., Kroemer, O., Peters, J.: Generalizing pouring actions between objects using warped parameters. In: Humanoids. (2014)
40. Schenck, C., Fox, D.: Visual closed-loop control for pouring liquids. In: ICRA. (2017)
41. Yamaguchi, A., Atkeson, C.G.: Differential dynamic programming with temporally decomposed dynamics. In: IEEE-RAS. (2015)
42. Kunze, L., Beetz, M.: Envisioning the qualitative effects of robot manipulation actions using simulation-based projections. Artificial Intelligence (2017)
43. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. (2016)
44. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: NIPS. (2014)

45. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR. (2009)